

Craypat OpenMP and MPI Metrics

**Heidi Poxon
Technical Lead, Performance Tools
Cray Inc.
heidi@cray.com**

NCCS workshop
ORNL
April 14-16, 2008



Heidi Poxon (heidi@cray.com) © Cray Inc.

OpenMP Performance Metrics Availability

- Available in **craypat/Apprentice2 4.2** and later
- craypat supports OpenMP from the following compilers
 - PGI, Pathscale
 - GNU support will be available in craypat 4.2.1
- Use of HW counters and OpenMP available in CNL 2.1

How to Collect OpenMP Performance Data

- Performance data collected by default for sampling
- Event traces
 - pat_build -g omp
 - pat_build -g omp_rtl
- User level API available to instrument OpenMP constructs
 - PGI 7.2 will automatically insert calls to trace points
 - Similar to Cray compiler on X2 systems
 - craypat 4.2.1 will recognize the trace points

OpenMP Performance Metrics

- Per-thread timings
- Overhead incurred at enter/exit of
 - parallel regions
 - worksharing constructs within parallel regions
- Load balance information across threads
- Sampling performance data without API
- Separate metrics for OpenMP runtime and OpenMP API calls

Trace Point API for OpenMP Constructs

- C functions

```
extern void PAT_omp_parallel_enter (void);
extern void PAT_omp_parallel_exit (void);
extern void PAT_omp_parallel_begin (void);
extern void PAT_omp_parallel_end (void);
extern void PAT_omp_loop_enter (void);
extern void PAT_omp_loop_exit (void);
extern void PAT_omp_sections_enter (void);
extern void PAT_omp_sections_exit (void);
extern void PAT_omp_section_begin (void);
extern void PAT_omp_section_end (void);
```

- Fortran subroutines have same names

OpenMP Data from pat_report

- Default view (no options needed to pat_report)
 - focus on where program is spending its time
 - shows imbalance across all threads
 - assumes all requested resources should be used
 - Highlights non-uniform imbalance across threads
 - Top threads got most of the work
 - Bottom threads got least of the work

Options for Data Display (pat_report -O ...)

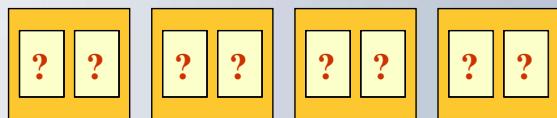
- profile_pe_th
 - Highlights PE imbalance
 - uses max for thread aggregation to avoid showing under-performers
 - aggregated thread data merged into PE data
- profile_th_pe
 - For each thread, show imbalance over PEs
 - Example: Load imbalance shown where thread 4 in each PE didn't get much work

MPI Metrics

- Obtained with:
 - pat_build -O apa
 - pat_build -g mpi
- Example metrics
 - Sent message statistics (default)
 - Bin statistics as total bytes
 - Bin statistics as counts of messages

MPI Rank Reorder

- MPI rank placement with environment variable



- Distributed placement
- SMP style placement
- Folded rank placement
- User provided rank file

MPI Rank Order Suggestions

- When to use?
 - Point-to-point communication consumes significant fraction of program time and load imbalance detected
- Available if MPI functions traced (-g mpi)
 - pat_build -O my_program.apa
- Information in resulting report
- Custom placement files automatically generated
- pat_report -O mpi_sm_rank_order (sent message)
- pat_report -O mpi_rank_order (user time)
- Add -s rank_grid_dim=N (N=leading grid dimension)

MPI Rank Order Suggestions (cont'd)

- See table notes in resulting report from pat_report
- Choose dual core or quad core suggestions from report
- Set MPICH_RANK_REORDER_METHOD environment variable
 - Set to numerical value or MPICH_RANK_ORDER file from pat_report

Example: -O mpi_sm_rank_order (sweep3d)

Notes for table 1:

To maximize the locality of point to point communication, choose and specify a Rank Order with small Max and Avg Sent Msg Total Bytes per node for the target number of cores per node.

To specify a Rank Order with a numerical value, set the environment variable `MPICH_RANK_REORDER_METHOD` to the given value.

To specify a Rank Order with a letter value 'x', set the environment variable `MPICH_RANK_REORDER_METHOD` to 3, and copy or link the file `MPICH_RANK_ORDER.x` to `MPICH_RANK_ORDER`.

Example – Swim 16 Processors

Dual core: Sent Msg Total Bytes per node

Rank Order	Max Total Bytes	Avg Total Bytes	Min Total Bytes	Max Node Ranks	Min Node Ranks
d	28736208	28736208	28736208	7,8	7,8
1	41054984	31815902	28736208	0,1	2,3
u	41054984	31815902	28736208	0,1	10,11
2	57472416	50288364	28736208	9,6	8,7
0	69791192	60552110	57472416	0,8	1,9

Quad core: Sent Msg Total Bytes per node

Rank Order	Max Total Bytes	Avg Total Bytes	Min Total Bytes	Max Node Ranks	Min Node Ranks
1	41054984	34895596	28736208	0,1,2,3	4,5,6,7
2	57472416	43104312	28736208	10,5,11,4	8,7,9,6
d	57472416	43104312	28736208	5,6,11,12	7,8,9,10
u	69791192	56447752	28736208	0,1,4,5	10,11,8,9
0	69791192	63631804	57472416	0,8,1,9	2,10,3,11

MPI Sync Time

- Determines if MPI ranks arrive at collectives together
- Separates potential load imbalance from data transfer
- Sync times reported by default if MPI functions traced (-O apa, -g mpi)

MPI + OpenMP? (some ideas)

- When does it pay to add OpenMP to my MPI code?
 - Only add OpenMP when code is network bound
 - Adding OpenMP to memory bound codes will most likely hurt performance rather than help it
 - Look at collective time, excluding sync time: this goes up as network becomes a problem
 - Look at point-to-point wait times: if these go up, network may be a problem